Subexponential Parameterized Algorithms for Bounded-Degree Connected Subgraph Problems on Planar Graphs

Ignasi Sau
Mascotte Project - CNRS/INRIA/UNSA (France) +
Applied Mathematics IV Department of UPC (Spain)
Dimitrios M. Thilikos
Department of Mathematics of National Capodistrian University of Athens (Greece)

10èmes Journées Graphes et Algorithmes (JGA)
Sophia Antipolis - November 7th, 2008

Outline of the talk

1. Preliminaries
   - FPT and subexponential algorithms
   - Branchwidth
   - Minors
   - Parameters

2. General framework to obtain subexponential algorithms
   - Bidimensionality
   - *Fast* dynamic programming

3. MAXIMUM d-DEGREE-BOUNDED CONNECTED SUBGRAPH (MDBCS$_d$)
   - Definition + example
   - Bidimensional behaviour
   - Dynamic programming techniques

1. Preliminaries

FPT and subexponential algorithms

Given a (NP-hard) problem with input of size n and a parameter k:

- A **fixed-parameter tractable** (FPT) algorithm runs in

$$f(k) \cdot n^{\mathcal{O}(1)}, \quad \text{for some function } f.$$

Examples: k-Vertex Cover, k-Longest Path.

- A **subexponential parameterized algorithm** is a FPT algo s.t.

$$f(k) = 2^{o(k)}.$$

- Typically $f(k) = 2^{\mathcal{O}(\sqrt{k})}$.

- The aim of this talk is to explain how to obtain **subexponential parameterized algorithms** for some NP-hard problems on **planar graphs**.

FPT and subexponential algorithms

Given a (NP-hard) problem with input of size n and a parameter k:

- A **fixed-parameter tractable** (FPT) algorithm runs in

$$f(k) \cdot n^{\mathcal{O}(1)}, \quad \text{for some function } f.$$

Examples: k-VERTEX COVER, k-LONGEST PATH.

- A **subexponential parameterized algorithm** is a FPT algo s.t.

$$f(k) = 2^{o(k)}.$$

- Typically $f(k) = 2^{\mathcal{O}(\sqrt{k})}$.

- The aim of this talk is to explain how to obtain **subexponential parameterized algorithms** for some NP-hard problems on **planar graphs**.

FPT and subexponential algorithms

Given a (NP-hard) problem with input of size n and a parameter k:

- A **fixed-parameter tractable** (FPT) algorithm runs in

$$f(k) \cdot n^{\mathcal{O}(1)}, \quad \text{for some function } f.$$

  Examples: k-VERTEX COVER, k-LONGEST PATH.

- A **subexponential parameterized algorithm** is a FPT algo s.t.

$$f(k) = 2^{o(k)}.$$

- Typically $f(k) = 2^{\mathcal{O}(\sqrt{k})}$.

- The aim of this talk is to explain how to obtain **subexponential parameterized algorithms** for some NP-hard problems on **planar graphs**.

FPT and subexponential algorithms

Given a (NP-hard) problem with input of size n and a parameter k:

- A **fixed-parameter tractable** (FPT) algorithm runs in

$$f(k) \cdot n^{\mathcal{O}(1)}, \quad \text{for some function } f.$$

Examples: k-VERTEX COVER, k-LONGEST PATH.

- A **subexponential parameterized algorithm** is a FPT algo s.t.

$$f(k) = 2^{o(k)}.$$

- Typically $f(k) = 2^{\mathcal{O}(\sqrt{k})}$.
- The aim of this talk is to explain how to obtain **subexponential parameterized algorithms** for some NP-hard problems on **planar graphs**.

FPT and subexponential algorithms

Given a (NP-hard) problem with input of size n and a parameter k:

- A **fixed-parameter tractable** (FPT) algorithm runs in

  $$f(k) \cdot n^{\mathcal{O}(1)}, \quad \text{for some function } f.$$

  Examples: k-VERTEX COVER, k-LONGEST PATH.

- A **subexponential parameterized algorithm** is a FPT algo s.t.

  $$f(k) = 2^{o(k)}.$$

- Typically $f(k) = 2^{\mathcal{O}(\sqrt{k})}$.

- The aim of this talk is to explain how to obtain **subexponential parameterized algorithms** for some NP-hard problems on **planar graphs**.

FPT and subexponential algorithms

Given a (NP-hard) problem with input of size n and a parameter k:

- A **fixed-parameter tractable** (FPT) algorithm runs in

  $$f(k) \cdot n^{\mathcal{O}(1)}, \quad \text{for some function } f.$$

  Examples: k-VERTEX COVER, k-LONGEST PATH.

- A **subexponential parameterized algorithm** is a FPT algo s.t.

  $$f(k) = 2^{o(k)}.$$

- Typically $f(k) = 2^{\mathcal{O}(\sqrt{k})}$.
- The aim of this talk is to explain how to obtain **subexponential parameterized algorithms** for some NP-hard problems on **planar graphs.**

Branchwidth

- A branch decomposition of a graph $G = (V, E)$ is tuple (T, μ) where:
  - T is a tree where all the internal nodes have degree 3.
  - μ is a bijection between the leaves of T and $E(G)$.
- Each edge $e \in T$ partitions $E(G)$ into two sets A_e and B_e.
- For each $e \in E(T)$, we define $\mathrm{mid}(e) = V(A_e) \cap V(B_e)$.
- The width of a branch decomposition is $\max_{e \in E(T)} |\mathrm{mid}(e)|$.
- The branchwidth of a graph G (denoted $\mathbf{bw}(G)$) is the minimum width over all branch decompositions of G:

$$\mathbf{bw}(G) = \min_{(T,\mu)} \max_{e \in E(T)} |\mathrm{mid}(e)|$$

Branchwidth

- A branch decomposition of a graph $G = (V, E)$ is tuple (T, μ) where:
  - T is a tree where all the internal nodes have degree 3.
  - μ is a bijection between the leaves of T and $E(G)$.
- Each edge $e \in T$ partitions $E(G)$ into two sets A_e and B_e.
- For each $e \in E(T)$, we define $\mathrm{mid}(e) = V(A_e) \cap V(B_e)$.
- The width of a branch decomposition is $\max_{e \in E(T)} |\mathrm{mid}(e)|$.
- The branchwidth of a graph G (denoted $\mathbf{bw}(G)$) is the minimum width over all branch decompositions of G:

$$\mathbf{bw}(G) = \min_{(T,\mu)} \max_{e \in E(T)} |\mathrm{mid}(e)|$$

Branchwidth

- A branch decomposition of a graph $G = (V, E)$ is tuple (T, μ) where:
  - T is a tree where all the internal nodes have degree 3.
  - μ is a bijection between the leaves of T and $E(G)$.
- Each edge $e \in T$ partitions $E(G)$ into two sets A_e and B_e.
- For each $e \in E(T)$, we define $\mathrm{mid}(e) = V(A_e) \cap V(B_e)$.
- The width of a branch decomposition is $\max_{e \in E(T)} |\mathrm{mid}(e)|$.
- The branchwidth of a graph G (denoted $\mathbf{bw}(G)$) is the minimum width over all branch decompositions of G:

$$\mathbf{bw}(G) = \min_{(T,\mu)} \max_{e \in E(T)} |\mathrm{mid}(e)|$$

Branchwidth

- A branch decomposition of a graph $G = (V, E)$ is tuple (T, μ) where:
    - T is a tree where all the internal nodes have degree 3.
    - μ is a bijection between the leaves of T and $E(G)$.
- Each edge $e \in T$ partitions $E(G)$ into two sets A_e and B_e.
- For each $e \in E(T)$, we define $\mathrm{mid}(e) = V(A_e) \cap V(B_e)$.
- The width of a branch decomposition is $\max_{e \in E(T)} |\mathrm{mid}(e)|$.
- The branchwidth of a graph G (denoted $\mathbf{bw}(G)$) is the minimum width over all branch decompositions of G:

$$\mathbf{bw}(G) = \min_{(T,\mu)} \max_{e \in E(T)} |\mathrm{mid}(e)|$$

Branchwidth

- A branch decomposition of a graph $G = (V, E)$ is tuple (T, μ) where:
  - T is a tree where all the internal nodes have degree 3.
  - μ is a bijection between the leaves of T and $E(G)$.
- Each edge $e \in T$ partitions $E(G)$ into two sets A_e and B_e.
- For each $e \in E(T)$, we define $\mathrm{mid}(e) = V(A_e) \cap V(B_e)$.
- The width of a branch decomposition is $\max_{e \in E(T)} |\mathrm{mid}(e)|$.
- The branchwidth of a graph G (denoted $\mathbf{bw}(G)$) is the minimum width over all branch decompositions of G:

$$\mathbf{bw}(G) = \min_{(T,\mu)} \max_{e \in E(T)} |\mathrm{mid}(e)|$$

Branchwidth

- A branch decomposition of a graph $G = (V, E)$ is tuple (T, μ) where:
  - T is a tree where all the internal nodes have degree 3.
  - μ is a bijection between the leaves of T and $E(G)$.
- Each edge $e \in T$ partitions $E(G)$ into two sets A_e and B_e.
- For each $e \in E(T)$, we define $\mathrm{mid}(e) = V(A_e) \cap V(B_e)$.
- The width of a branch decomposition is $\max_{e \in E(T)} |\mathrm{mid}(e)|$.
- The branchwidth of a graph G (denoted $\mathbf{bw}(G)$) is the minimum width over all branch decompositions of G:

$$\mathbf{bw}(G) = \min_{(T,\mu)} \max_{e \in E(T)} |\mathrm{mid}(e)|$$

Graph minors

- H is a contraction of G ($H \preceq_c G$) if H occurs from G after applying a series of edge contractions.
- H is a minor of G ($H \preceq_m G$) if H is the contraction of some subgraph of G.
- A graph class $\mathcal{G}$ is minor closed if every minor of a graph in $\mathcal{G}$ is again in $\mathcal{G}$.
- A graph class $\mathcal{G}$ is H-minor-free (or, excludes H as a minor) if no graph in $\mathcal{G}$ contains H as a minor.

Graph minors

- H is a contraction of G ($H \preceq_c G$) if H occurs from G after applying a series of edge contractions.
- H is a minor of G ($H \preceq_m G$) if H is the contraction of some subgraph of G.
- A graph class $\mathcal{G}$ is minor closed if every minor of a graph in $\mathcal{G}$ is again in $\mathcal{G}$.
- A graph class $\mathcal{G}$ is H-minor-free (or, excludes H as a minor) if no graph in $\mathcal{G}$ contains H as a minor.

Graph minors

- H is a contraction of G ($H \preceq_c G$) if H occurs from G after applying a series of edge contractions.
- H is a minor of G ($H \preceq_m G$) if H is the contraction of some subgraph of G.
- A graph class $\mathcal{G}$ is minor closed if every minor of a graph in $\mathcal{G}$ is again in $\mathcal{G}$.
- A graph class $\mathcal{G}$ is H-minor-free (or, excludes H as a minor) if no graph in $\mathcal{G}$ contains H as a minor.

Graph minors

- H is a contraction of G ($H \preceq_c G$) if H occurs from G after applying a series of edge contractions.
- H is a minor of G ($H \preceq_m G$) if H is the contraction of some subgraph of G.
- A graph class $\mathcal{G}$ is minor closed if every minor of a graph in $\mathcal{G}$ is again in $\mathcal{G}$.
- A graph class $\mathcal{G}$ is H-minor-free (or, excludes H as a minor) if no graph in $\mathcal{G}$ contains H as a minor.

Graph Minors Theorem

- **Robertson** and **Seymour** (1986-2004):

Theorem (Graphs Minors Theorem)

Graphs are well-quasi-ordered by the minor relation $\preceq_m$.

- **Consequence**: every minor closed graph class $\mathcal{G}$ has a finite set of minimal excluded minors.

- **Algorithmic Consequence**: Membership testing for any minor closed graph class $\mathcal{G}$ can be done in polynomial time ($\mathcal{O}(n^3)$).

Graph Minors Theorem

- **Robertson** and **Seymour** (1986-2004):

Theorem (Graphs Minors Theorem)

Graphs are well-quasi-ordered by the minor relation $\preceq_m$.

- **Consequence**: every minor closed graph class $\mathcal{G}$ has a finite set of minimal excluded minors.

- **Algorithmic Consequence**: Membership testing for any minor closed graph class $\mathcal{G}$ can be done in polynomial time ($\mathcal{O}(n^3)$).

Graph Minors Theorem

- **Robertson** and **Seymour** (1986-2004):

Theorem (Graphs Minors Theorem)

Graphs are well-quasi-ordered by the minor relation $\preceq_m$.

- **Consequence**: every minor closed graph class $\mathcal{G}$ has a finite set of minimal excluded minors.

- **Algorithmic Consequence**: Membership testing for any minor closed graph class $\mathcal{G}$ can be done in polynomial time ($\mathcal{O}(n^3)$).

Parameters

- A parameter **P** is any function mapping graphs to non-negative integers:

$$\mathbf{P} : \mathcal{G} \rightarrow \mathbb{N}^+$$

- *Examples*: Size of a minimum vertex cover, size of a maximum clique, ...

- The parameterized problem associated with **P** asks, for some fixed k, whether $\mathbf{P}(G) \geq k$ for a given graph G.

- We say that a parameter **P** is minor closed if for every graph H,

$$H \preceq_m G \;\; \Rightarrow \;\; \mathbf{P}(H) \leq \mathbf{P}(G).$$

Parameters

- A parameter **P** is any function mapping graphs to non-negative integers:

$$\mathbf{P} : \mathcal{G} \rightarrow \mathbb{N}^{+}$$

- *Examples*: Size of a minimum vertex cover, size of a maximum clique, ...

- The parameterized problem associated with **P** asks, for some fixed k, whether $\mathbf{P}(G) \geq k$ for a given graph G.

- We say that a parameter **P** is minor closed if for every graph H,

$$H \preceq_m G \;\; \Rightarrow \;\; \mathbf{P}(H) \leq \mathbf{P}(G).$$

Parameters

- A parameter **P** is any function mapping graphs to non-negative integers:

$$\mathbf{P} : \mathcal{G} \rightarrow \mathbb{N}^+$$

- *Examples*: Size of a minimum vertex cover, size of a maximum clique, ...
- The parameterized problem associated with **P** asks, for some fixed k, whether $\mathbf{P}(G) \geq k$ for a given graph G.
- We say that a parameter **P** is minor closed if for every graph H,

$$H \preceq_m G \quad \Rightarrow \quad \mathbf{P}(H) \leq \mathbf{P}(G).$$

Parameters

- A parameter **P** is any function mapping graphs to non-negative integers:

$$\mathbf{P} : \mathcal{G} \rightarrow \mathbb{N}^{+}$$

- *Examples*: Size of a minimum vertex cover, size of a maximum clique, ...
- The parameterized problem associated with **P** asks, for some fixed k, whether $\mathbf{P}(G) \geq k$ for a given graph G.
- We say that a parameter **P** is minor closed if for every graph H,

$$H \preceq_m G \quad \Rightarrow \quad \mathbf{P}(H) \leq \mathbf{P}(G).$$

An algorithmic consequence of the Graph Minors Theorem

- Every minor closed parameterized problem has an

$$\mathcal{O}(f(k) \cdot n^{\mathcal{O}(1)})$$

step algorithm.

- **Problem**: $f(k)$ is unknown or huge!
- **Question**: How and when can we improve $f(k)$ above?
- **Question**: When can $f(k)$ be a subexponential function?

An algorithmic consequence of the Graph Minors Theorem

- Every minor closed parameterized problem has an

$$\mathcal{O}(f(k) \cdot n^{\mathcal{O}(1)})$$

step algorithm.

- **Problem**: $f(k)$ is unknown or huge!
- **Question**: How and when can we improve $f(k)$ above?
- **Question**: When can $f(k)$ be a subexponential function?

An algorithmic consequence of the Graph Minors Theorem

- Every minor closed parameterized problem has an

$$\mathcal{O}(f(k) \cdot n^{\mathcal{O}(1)})$$

step algorithm.

- **Problem**: $f(k)$ is unknown or huge!
- **Question**: How and when can we improve $f(k)$ above?
- **Question**: When can $f(k)$ be a subexponential function?

An algorithmic consequence of the Graph Minors Theorem

- Every minor closed parameterized problem has an

$$\mathcal{O}(f(k) \cdot n^{\mathcal{O}(1)})$$

  step algorithm.

- **Problem**: $f(k)$ is unknown or huge!
- **Question**: How and when can we improve $f(k)$ above?
- **Question**: When can $f(k)$ be a subexponential function?

2. General framework to obtain subexponential parameterized algorithms

Subexponential parameterized algorithms on planar graphs

- [J. Alber, H. L. Bodlaender, H. Fernau, T. Kloks, R. Niedermeier. SWAT'00, Algorithmica 2002]
  - $\mathcal{O}(c^{\sqrt{k}}n)$ algorithm for k-DOMINATING SET on planar graphs.
  - First non-trivial result for an NP-hard FPT problem with sublinear exponent.
- Other references:
  - [Alber, Fernau, and Niedermeier. J. Algorithms 2004]
  - [M. S. Chang, T. Kloks, and C. M. Lee. WG'01]
  - [Gutin, Kloks, Lee, and Yeo. J. Comput. System Sci. 2005]
  - [Fernau. MFCS' 04]
  - [Kanj and L. Perković. MFCS' 02]

Subexponential parameterized algorithms on planar graphs

- [J. Alber, H. L. Bodlaender, H. Fernau, T. Kloks, R. Niedermeier. SWAT'00, Algorithmica 2002]
  - $\mathcal{O}(c^{\sqrt{k}}n)$ algorithm for k-DOMINATING SET on planar graphs.
  - First non-trivial result for an NP-hard FPT problem with sublinear exponent.
- Other references:
  - [Alber, Fernau, and Niedermeier. J. Algorithms 2004]
  - [M. S. Chang, T. Kloks, and C. M. Lee. WG'01]
  - [Gutin, Kloks, Lee, and Yeo. J. Comput. System Sci. 2005]
  - [Fernau. MFCS' 04]
  - [Kanj and L. Perković. MFCS' 02]

Subexponential parameterized algorithms on planar graphs

- [J. Alber, H. L. Bodlaender, H. Fernau, T. Kloks, R. Niedermeier. SWAT'00, Algorithmica 2002]
  - $\mathcal{O}(c^{\sqrt{k}} n)$ algorithm for k-DOMINATING SET on planar graphs.
  - First non-trivial result for an NP-hard FPT problem with sublinear exponent.
- Other references:
  - [Alber, Fernau, and Niedermeier. J. Algorithms 2004]
  - [M. S. Chang, T. Kloks, and C. M. Lee. WG'01]
  - [Gutin, Kloks, Lee, and Yeo. J. Comput. System Sci. 2005]
  - [Fernau. MFCS' 04]
  - [Kanj and L. Perković. MFCS' 02]

Subexponential parameterized algorithms on planar graphs

- [J. Alber, H. L. Bodlaender, H. Fernau, T. Kloks, R. Niedermeier. SWAT'00, Algorithmica 2002]
  - $\mathcal{O}(c^{\sqrt{k}}n)$ algorithm for k-DOMINATING SET on planar graphs.
  - First non-trivial result for an NP-hard FPT problem with sublinear exponent.
- Other references:
  - [Alber, Fernau, and Niedermeier. J. Algorithms 2004]
  - [M. S. Chang, T. Kloks, and C. M. Lee. WG'01]
  - [Gutin, Kloks, Lee, and Yeo. J. Comput. System Sci. 2005]
  - [Fernau. MFCS' 04]
  - [Kanj and L. Perković. MFCS' 02]

General idea / *meta-algorithmic* framework

Given a parameter **P** defined in a graph class $\mathcal{G}$:

(A) Combinatorial bounds via Graph Minor theorems:

For every graph $G \in \mathcal{G}$, $\mathbf{bw}(G) \leq \alpha \cdot \sqrt{\mathbf{P}(G)} + \mathcal{O}(1)$

- **Bidimensionality.**
  [E.D. Demaine, F.V. Fomin, M.T. Hajiaghayi, D.M. Thilikos. SODA'04, J.ACM'05]

(B) Dynamic programming which uses graph structure:

For every graph $G \in \mathcal{G}$ and given an optimal branch decomposition (T, μ) of G, the value of $\mathbf{P}(G)$ can be computed in $f(\mathbf{bw}(G)) \cdot n^{\mathcal{O}(1)}$ steps.

- **Catalan structures.**
  [F. Dorn, F.V. Fomin, D.M. Thilikos. ICALP'07]
  [F. Dorn, F.V. Fomin, D.M. Thilikos. SODA'08]

General idea / *meta-algorithmic* framework

Given a parameter **P** defined in a graph class $\mathcal{G}$:

(A) Combinatorial bounds via Graph Minor theorems:

$$\boxed{\text{For every graph } G \in \mathcal{G},\ \mathbf{bw}(G) \leq \alpha \cdot \sqrt{\mathbf{P}(G)} + \mathcal{O}(1)}$$

- **Bidimensionality.**
  [E.D. Demaine, F.V. Fomin, M.T. Hajiaghayi, D.M. Thilikos. SODA'04, J.ACM'05]

(B) Dynamic programming which uses graph structure:

For every graph $G \in \mathcal{G}$ and given an optimal branch decomposition (T, μ) of G, the value of $\mathbf{P}(G)$ can be computed in $f(\mathbf{bw}(G)) \cdot n^{\mathcal{O}(1)}$ steps.

- **Catalan structures.**
  [F. Dorn, F.V. Fomin, D.M. Thilikos. ICALP'07]
  [F. Dorn, F.V. Fomin, D.M. Thilikos. SODA'08]

General idea / *meta-algorithmic* framework

Given a parameter **P** defined in a graph class $\mathcal{G}$:

(A) Combinatorial bounds via Graph Minor theorems:

$$\boxed{\text{For every graph } G \in \mathcal{G},\ \mathbf{bw}(G) \leq \alpha \cdot \sqrt{\mathbf{P}(G)} + \mathcal{O}(1)}$$

- **Bidimensionality.**
  [E.D. Demaine, F.V. Fomin, M.T. Hajiaghayi, D.M. Thilikos. SODA'04, J.ACM'05]

(B) Dynamic programming which uses graph structure:

For every graph $G \in \mathcal{G}$ and given an optimal branch decomposition (T, μ) of G, the value of $\mathbf{P}(G)$ can be computed in $f(\mathbf{bw}(G)) \cdot n^{\mathcal{O}(1)}$ steps.

- **Catalan structures.**
  [F. Dorn, F.V. Fomin, D.M. Thilikos. ICALP'07]
  [F. Dorn, F.V. Fomin, D.M. Thilikos. SODA'08]

General idea / *meta-algorithmic* framework

Given a parameter **P** defined in a graph class $\mathcal{G}$:

(A) Combinatorial bounds via Graph Minor theorems:

For every graph $G \in \mathcal{G}$, $\mathbf{bw}(G) \leq \alpha \cdot \sqrt{\mathbf{P}(G)} + \mathcal{O}(1)$

- **Bidimensionality.**
  [E.D. Demaine, F.V. Fomin, M.T. Hajiaghayi, D.M. Thilikos. SODA'04, J.ACM'05]

(B) Dynamic programming which uses graph structure:

For every graph $G \in \mathcal{G}$ and given an optimal branch decomposition (T, μ) of G, the value of $\mathbf{P}(G)$ can be computed in $f(\mathbf{bw}(G)) \cdot n^{\mathcal{O}(1)}$ steps.

- **Catalan structures.**
  [F. Dorn, F.V. Fomin, D.M. Thilikos. ICALP'07]
  [F. Dorn, F.V. Fomin, D.M. Thilikos. SODA'08]

General idea / *meta-algorithmic* framework

Given a parameter **P** defined in a graph class $\mathcal{G}$:

(A) Combinatorial bounds via Graph Minor theorems:

For every graph $G \in \mathcal{G}$, $\mathbf{bw}(G) \leq \alpha \cdot \sqrt{\mathbf{P}(G)} + \mathcal{O}(1)$

- **Bidimensionality.**
  [E.D. Demaine, F.V. Fomin, M.T. Hajiaghayi, D.M. Thilikos. SODA'04, J.ACM'05]

(B) Dynamic programming which uses graph structure:

For every graph $G \in \mathcal{G}$ and given an optimal branch decomposition (T, μ) of G, the value of $\mathbf{P}(G)$ can be computed in $f(\mathbf{bw}(G)) \cdot n^{\mathcal{O}(1)}$ steps.

- **Catalan structures.**
  [F. Dorn, F.V. Fomin, D.M. Thilikos. ICALP'07]
  [F. Dorn, F.V. Fomin, D.M. Thilikos. SODA'08]

Explicit algorithm

(A) For every graph $G \in \mathcal{G}$, $\mathbf{bw}(G) \leq \alpha \cdot \sqrt{\mathbf{P}(G)} + \mathcal{O}(1)$

(B) For every graph $G \in \mathcal{G}$ and given an optimal branch decomposition (T, μ) of G, the value of $\mathbf{P}(G)$ can be computed in $f(\mathbf{bw}(G)) \cdot n^{\mathcal{O}(1)}$ steps.

Case 1: If $\mathbf{bw}(G) > \alpha \cdot \sqrt{k}$, then by **(A)**:

$$\alpha \cdot \sqrt{k} < \mathbf{bw}(G) \leq \alpha \cdot \sqrt{\mathbf{P}(G)} + \mathcal{O}(1) \quad \Rightarrow \quad \mathbf{P}(G) \geq k.$$

Case 2: Otherwise ($\mathbf{bw}(G) \leq \alpha \cdot \sqrt{k}$) by **(B)**, $\mathbf{P}(G)$ can be computed in $f(\alpha \cdot \sqrt{k}) \cdot n^{\mathcal{O}(1)}$ steps.

- If $f(\ell) = 2^{\mathcal{O}(\ell)}$, this strategy yields an exact algorithm with running time $2^{\mathcal{O}(\sqrt{k})} \cdot n^{\mathcal{O}(1)} \quad \rightarrow \quad$ **subexponential!**

- **Note**: we must add $\mathcal{O}(n^3)$ to compute an optimal branch decomposition of a planar graph.

Explicit algorithm

(A) For every graph $G \in \mathcal{G}$, $\mathbf{bw}(G) \leq \alpha \cdot \sqrt{\mathbf{P}(G)} + \mathcal{O}(1)$

(B) For every graph $G \in \mathcal{G}$ and given an optimal branch decomposition (T, μ) of G, the value of $\mathbf{P}(G)$ can be computed in $f(\mathbf{bw}(G)) \cdot n^{\mathcal{O}(1)}$ steps.

Case 1: If $\mathbf{bw}(G) > \alpha \cdot \sqrt{k}$, then by **(A)**:

$$\alpha \cdot \sqrt{k} < \mathbf{bw}(G) \leq \alpha \cdot \sqrt{\mathbf{P}(G)} + \mathcal{O}(1) \quad \Rightarrow \quad \mathbf{P}(G) \geq k.$$

Case 2: Otherwise ($\mathbf{bw}(G) \leq \alpha \cdot \sqrt{k}$) by **(B)**, $\mathbf{P}(G)$ can be computed in $f(\alpha \cdot \sqrt{k}) \cdot n^{\mathcal{O}(1)}$ steps.

- If $f(\ell) = 2^{\mathcal{O}(\ell)}$, this strategy yields an exact algorithm with running time $2^{\mathcal{O}(\sqrt{k})} \cdot n^{\mathcal{O}(1)} \quad \rightarrow \quad$ **subexponential!**

- **Note**: we must add $\mathcal{O}(n^3)$ to compute an optimal branch decomposition of a planar graph.

Explicit algorithm

(A) For every graph $G \in \mathcal{G}$, $\mathbf{bw}(G) \leq \alpha \cdot \sqrt{\mathbf{P}(G)} + \mathcal{O}(1)$

(B) For every graph $G \in \mathcal{G}$ and given an optimal branch decomposition (T, μ) of G, the value of $\mathbf{P}(G)$ can be computed in $f(\mathbf{bw}(G)) \cdot n^{\mathcal{O}(1)}$ steps.

Case 1: If $\mathbf{bw}(G) > \alpha \cdot \sqrt{k}$, then by **(A)**:

$$\alpha \cdot \sqrt{k} < \mathbf{bw}(G) \leq \alpha \cdot \sqrt{\mathbf{P}(G)} + \mathcal{O}(1) \quad \Rightarrow \quad \mathbf{P}(G) \geq k.$$

Case 2: Otherwise ($\mathbf{bw}(G) \leq \alpha \cdot \sqrt{k}$) by **(B)**, $\mathbf{P}(G)$ can be computed in $f(\alpha \cdot \sqrt{k}) \cdot n^{\mathcal{O}(1)}$ steps.

- If $f(\ell) = 2^{\mathcal{O}(\ell)}$, this strategy yields an exact algorithm with running time $2^{\mathcal{O}(\sqrt{k})} \cdot n^{\mathcal{O}(1)}$ $\rightarrow$ **subexponential!**

- **Note**: we must add $\mathcal{O}(n^3)$ to compute an optimal branch decomposition of a planar graph.

Explicit algorithm

(A) For every graph $G \in \mathcal{G}$, $\mathbf{bw}(G) \leq \alpha \cdot \sqrt{\mathbf{P}(G)} + \mathcal{O}(1)$

(B) For every graph $G \in \mathcal{G}$ and given an optimal branch decomposition (T, μ) of G, the value of $\mathbf{P}(G)$ can be computed in $f(\mathbf{bw}(G)) \cdot n^{\mathcal{O}(1)}$ steps.

Case 1: If $\mathbf{bw}(G) > \alpha \cdot \sqrt{k}$, then by **(A)**:

$$\alpha \cdot \sqrt{k} < \mathbf{bw}(G) \leq \alpha \cdot \sqrt{\mathbf{P}(G)} + \mathcal{O}(1) \quad \Rightarrow \quad \mathbf{P}(G) \geq k.$$

Case 2: Otherwise ($\mathbf{bw}(G) \leq \alpha \cdot \sqrt{k}$) by **(B)**, $\mathbf{P}(G)$ can be computed in $f(\alpha \cdot \sqrt{k}) \cdot n^{\mathcal{O}(1)}$ steps.

- If $f(\ell) = 2^{\mathcal{O}(\ell)}$, this strategy yields an exact algorithm with running time $2^{\mathcal{O}(\sqrt{k})} \cdot n^{\mathcal{O}(1)}$ $\rightarrow$ **subexponential!**
- **Note**: we must add $\mathcal{O}(n^3)$ to compute an optimal branch decomposition of a planar graph.

Explicit algorithm

(A) For every graph $G \in \mathcal{G}$, $\mathbf{bw}(G) \leq \alpha \cdot \sqrt{\mathbf{P}(G)} + \mathcal{O}(1)$

(B) For every graph $G \in \mathcal{G}$ and given an optimal branch decomposition (T, μ) of G, the value of $\mathbf{P}(G)$ can be computed in $f(\mathbf{bw}(G)) \cdot n^{\mathcal{O}(1)}$ steps.

Case 1: If $\mathbf{bw}(G) > \alpha \cdot \sqrt{k}$, then by **(A)**:

$$\alpha \cdot \sqrt{k} < \mathbf{bw}(G) \leq \alpha \cdot \sqrt{\mathbf{P}(G)} + \mathcal{O}(1) \quad \Rightarrow \quad \mathbf{P}(G) \geq k.$$

Case 2: Otherwise ($\mathbf{bw}(G) \leq \alpha \cdot \sqrt{k}$) by **(B)**, $\mathbf{P}(G)$ can be computed in $f(\alpha \cdot \sqrt{k}) \cdot n^{\mathcal{O}(1)}$ steps.

- If $f(\ell) = 2^{\mathcal{O}(\ell)}$, this strategy yields an exact algorithm with running time $2^{\mathcal{O}(\sqrt{k})} \cdot n^{\mathcal{O}(1)} \quad \rightarrow \quad$ **subexponential!**

- **Note**: we must add $\mathcal{O}(n^3)$ to compute an optimal branch decomposition of a planar graph.

Explicit algorithm

(A) For every graph $G \in \mathcal{G}$, $\mathbf{bw}(G) \leq \alpha \cdot \sqrt{\mathbf{P}(G)} + \mathcal{O}(1)$

(B) For every graph $G \in \mathcal{G}$ and given an optimal branch decomposition (T, μ) of G, the value of $\mathbf{P}(G)$ can be computed in $f(\mathbf{bw}(G)) \cdot n^{\mathcal{O}(1)}$ steps.

Case 1: If $\mathbf{bw}(G) > \alpha \cdot \sqrt{k}$, then by **(A)**:

$$\alpha \cdot \sqrt{k} < \mathbf{bw}(G) \leq \alpha \cdot \sqrt{\mathbf{P}(G)} + \mathcal{O}(1) \quad \Rightarrow \quad \mathbf{P}(G) \geq k.$$

Case 2: Otherwise ($\mathbf{bw}(G) \leq \alpha \cdot \sqrt{k}$) by **(B)**, $\mathbf{P}(G)$ can be computed in $f(\alpha \cdot \sqrt{k}) \cdot n^{\mathcal{O}(1)}$ steps.

- If $f(\ell) = 2^{\mathcal{O}(\ell)}$, this strategy yields an exact algorithm with running time $2^{\mathcal{O}(\sqrt{k})} \cdot n^{\mathcal{O}(1)} \quad \rightarrow \quad$ **subexponential!**

- **Note**: we must add $\mathcal{O}(n^3)$ to compute an optimal branch decomposition of a planar graph.

3. MAXIMUM d-DEGREE-BOUNDED CONNECTED SUBGRAPH

Definition of the problem: MDBCS$_d$

- **MAXIMUM *d*-DEGREE-BOUNDED CONNECTED SUBGRAPH:**

  Input:
  - an undirected graph $G = (V, E)$,
  - an integer $d \geq 2$, and
  - a weight function $w : E \to \mathbb{R}^+$.

  Output:
  a subset of edges $E' \subseteq E$ such that $G' = G[E']$
  - is **connected**,
  - $\Delta(G') \leq d$,
  - and maximising $\sum_{e \in E'} w(e)$.

- It is one of the classical **NP**-hard problems of *[Garey and Johnson. Computers and Intractability, 1979]*
- If the output subgraph is not required to be connected, the problem is in **P** for any *d* (using matching techniques).
- For *fixed* $d = 2$ it is the LONGEST PATH (OR CYCLE).

Definition of the problem: MDBCS$_d$

- **MAXIMUM d-DEGREE-BOUNDED CONNECTED SUBGRAPH:**

  Input:
  - an undirected graph $G = (V, E)$,
  - an integer $d \geq 2$, and
  - a weight function $w : E \to \mathbb{R}^+$.

  Output:
  a subset of edges $E' \subseteq E$ such that $G' = G[E']$
  - is **connected**,
  - $\Delta(G') \leq d$,
  - and maximising $\sum_{e \in E'} w(e)$.

- It is one of the classical **NP**-hard problems of *[Garey and Johnson. Computers and Intractability, 1979]*
- If the output subgraph is not required to be connected, the problem is in **P** for any d (using matching techniques).
- For *fixed* $d = 2$ it is the LONGEST PATH (OR CYCLE).

Definition of the problem: MDBCS$_d$

- **MAXIMUM d-DEGREE-BOUNDED CONNECTED SUBGRAPH:**

  Input:
  - an undirected graph $G = (V, E)$,
  - an integer $d \geq 2$, and
  - a weight function $w : E \rightarrow \mathbb{R}^+$.

  Output:
  a subset of edges $E' \subseteq E$ such that $G' = G[E']$
  - is **connected**,
  - $\Delta(G') \leq d$,
  - and maximising $\sum_{e \in E'} w(e)$.
- It is one of the classical **NP**-hard problems of *[Garey and Johnson. Computers and Intractability, 1979]*
- If the output subgraph is not required to be connected, the problem is in **P** for any d (using matching techniques).
- For *fixed* $d = 2$ it is the LONGEST PATH (OR CYCLE).

Definition of the problem: MDBCS$_d$

- **MAXIMUM d-DEGREE-BOUNDED CONNECTED SUBGRAPH:**

  Input:
  - an undirected graph $G = (V, E)$,
  - an integer $d \geq 2$, and
  - a weight function $w : E \to \mathbb{R}^+$.

  Output:
  a subset of edges $E' \subseteq E$ such that $G' = G[E']$
  - is **connected**,
  - $\Delta(G') \leq d$,
  - and maximising $\sum_{e \in E'} w(e)$.

- It is one of the classical **NP**-hard problems of *[Garey and Johnson. Computers and Intractability, 1979]*
- If the output subgraph is not required to be connected, the problem is in **P** for any d (using matching techniques).
- For *fixed* $d = 2$ it is the LONGEST PATH (OR CYCLE).

Definition of the problem: MDBCS$_d$

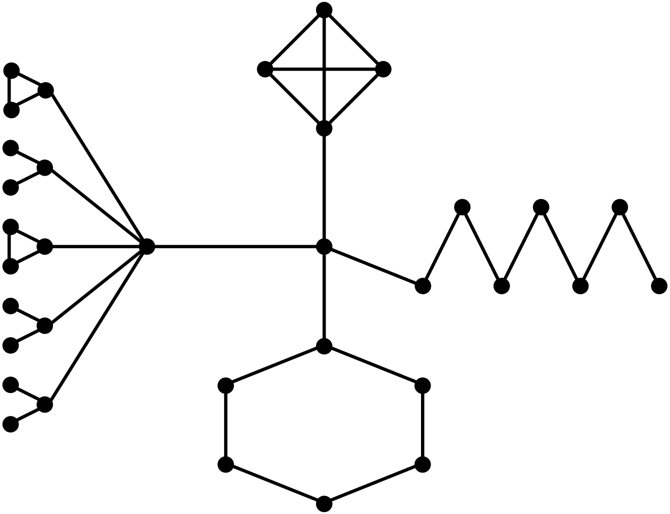
- **MAXIMUM d-DEGREE-BOUNDED CONNECTED SUBGRAPH:**

  Input:
  - an undirected graph $G = (V, E)$,
  - an integer $d \geq 2$, and
  - a weight function $w : E \rightarrow \mathbb{R}^+$.

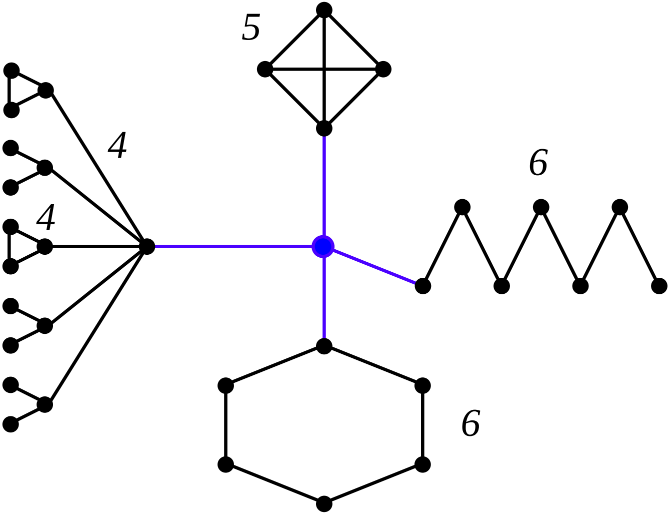
  Output:
  a subset of edges $E' \subseteq E$ such that $G' = G[E']$
  - is **connected**,
  - $\Delta(G') \leq d$,
  - and maximising $\sum_{e \in E'} w(e)$.

- It is one of the classical **NP**-hard problems of *[Garey and Johnson. Computers and Intractability, 1979]*
- If the output subgraph is not required to be connected, the problem is in **P** for any d (using matching techniques).
- For *fixed* $d = 2$ it is the **LONGEST PATH (OR CYCLE)**.

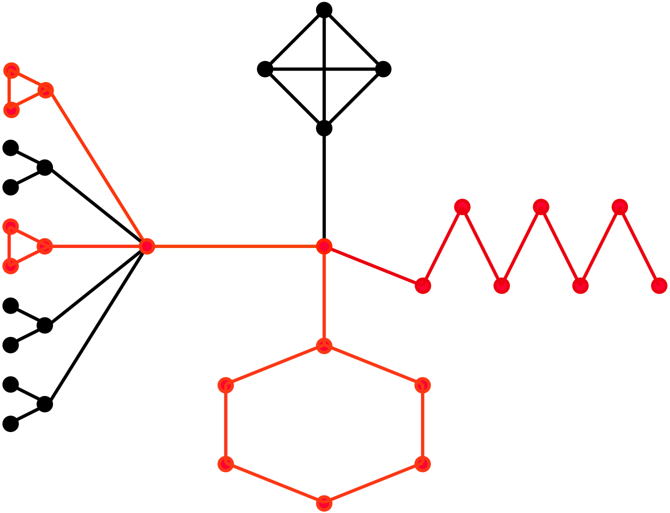
Example with $d = 3$, $\omega(e) = 1$ for all $e \in E(G)$



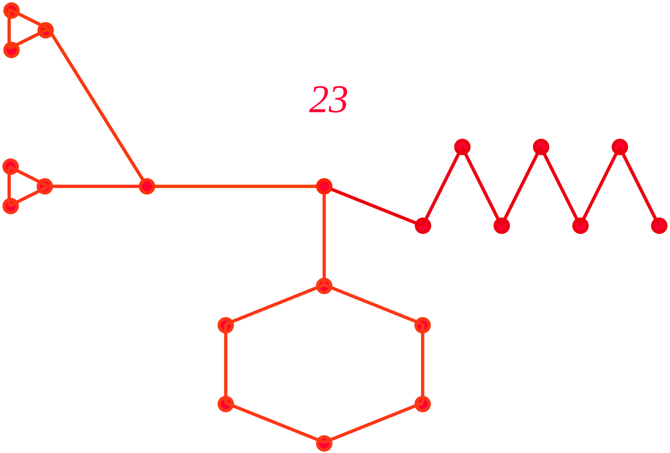
Example with $d = 3$ (II)



Example with $d = 3$ (III)



Example with $d = 3$ (IV)



State of the art

Case $d = 2$ (LONGEST PATH):

- **Approximation algorithms**:
  $\mathcal{O}\left(\frac{n}{\log n}\right)$-approximation, using the **color-coding** method.
  [N. Alon, R. Yuster and U. Zwick. STOC'94].
  $\mathcal{O}\left(n\left(\frac{\log\log n}{\log n}\right)^2\right)$-approximation.
  [A. Björklund and T. Husfeldt. SIAM J. Computing'03].

- **Hardness results**:
  It does not accept *any* constant-factor approximation.
  [D. Karger, R. Motwani, and G. Ramkumar. Algorithmica'97].

State of the art

Case $d = 2$ (LONGEST PATH):

- **Approximation algorithms**:
  $\mathcal{O}\left(\frac{n}{\log n}\right)$-approximation, using the **color-coding** method.
  [N. Alon, R. Yuster and U. Zwick. STOC'94].
  $\mathcal{O}\left(n\left(\frac{\log\log n}{\log n}\right)^2\right)$-approximation.
  [A. Björklund and T. Husfeldt. SIAM J. Computing'03].

- **Hardness results**:
  It does not accept *any* constant-factor approximation.
  [D. Karger, R. Motwani, and G. Ramkumar. Algorithmica'97].

State of the art

Case $d = 2$ (LONGEST PATH):

- **Approximation algorithms**:
  $\mathcal{O}\left(\frac{n}{\log n}\right)$-approximation, using the **color-coding** method.
  [N. Alon, R. Yuster and U. Zwick. STOC'94].
  $\mathcal{O}\left(n\left(\frac{\log\log n}{\log n}\right)^2\right)$-approximation.
  [A. Björklund and T. Husfeldt. SIAM J. Computing'03].
- **Hardness results**:
  It does not accept *any* constant-factor approximation.
  [D. Karger, R. Motwani, and G. Ramkumar. Algorithmica'97].

State of the art (II)

Case $d \geq 2$

[O. Amini, D. Peleg, S. Pérennes, I. S., S. Saurabh. ALGO/WAOA'08]:

- **Approximation algorithms** ($n = |V(G)|$, $m = |E(G)|$):
  - $\min\{\frac{n}{2}, \frac{m}{d}\}$-approximation algorithm for **weighted** graphs.
  - $\min\{\frac{m}{\log n}, \frac{nd}{2\log n}\}$-approximation algorithm for **unweighted** graphs, using *color coding*.
  - when G **accepts a low-degree spanning tree**, in terms of d, then MDBCS$_d$ can be approximated within a **small constant factor**.
- **Hardness results**:
  - For each fixed $d \geq 2$, MDBCS$_d$ does not accept *any* **constant-factor approximation** in general graphs.

State of the art (II)

Case $d \geq 2$
[O. Amini, D. Peleg, S. Pérennes, I. S., S. Saurabh. ALGO/WAOA'08]:

- **Approximation algorithms** ($n = |V(G)|$, $m = |E(G)|$):
  - $\min\{\frac{n}{2}, \frac{m}{d}\}$-approximation algorithm for **weighted** graphs.
  - $\min\{\frac{m}{\log n}, \frac{nd}{2 \log n}\}$-approximation algorithm for **unweighted** graphs, using *color coding*.
  - when G **accepts a low-degree spanning tree**, in terms of d, then MDBCS$_d$ can be approximated within a **small constant factor**.
- **Hardness results**:
  - For each fixed $d \geq 2$, MDBCS$_d$ does not accept ***any*** **constant-factor approximation** in general graphs.

State of the art (II)

Case $d \geq 2$
[O. Amini, D. Peleg, S. Pérennes, I. S., S. Saurabh. ALGO/WAOA'08]:

- **Approximation algorithms** ($n = |V(G)|$, $m = |E(G)|$):
  - $\min\{\frac{n}{2}, \frac{m}{d}\}$-approximation algorithm for **weighted** graphs.
  - $\min\{\frac{m}{\log n}, \frac{nd}{2\log n}\}$-approximation algorithm for **unweighted** graphs, using *color coding*.
  - when G **accepts a low-degree spanning tree**, in terms of d, then MDBCS$_d$ can be approximated within a **small constant factor**.

- **Hardness results**:
  - For each fixed $d \geq 2$, MDBCS$_d$ does not accept ***any*** **constant-factor approximation** in general graphs.

State of the art (II)

Case $d \geq 2$
[O. Amini, D. Peleg, S. Pérennes, I. S., S. Saurabh. ALGO/WAOA'08]:

- **Approximation algorithms** ($n = |V(G)|$, $m = |E(G)|$):
  - $\min\{\frac{n}{2}, \frac{m}{d}\}$-approximation algorithm for **weighted** graphs.
  - $\min\{\frac{m}{\log n}, \frac{nd}{2\log n}\}$-approximation algorithm for **unweighted** graphs, using *color coding*.
  - when G **accepts a low-degree spanning tree**, in terms of d, then MDBCS_d can be approximated within a **small constant factor**.
- **Hardness results**:
  - For each fixed $d \geq 2$, MDBCS_d does not accept ***any*** **constant-factor approximation** in general graphs.

Let us apply the general strategy...

We define the following **parameter** on a **planar** graph G:

$\mathbf{mdbcs}_d(G) = \max\{|E(H)| \mid H \subseteq G \wedge H \text{ is connected } \wedge \Delta(H) \leq d\}$.

(we focus on the **unweighted** version of the problem)

We distinguish two cases according to **bw**(G):

(A) If **bw**(G) is ***big*** $(> \alpha \cdot \sqrt{k})$:
we must exhibit a ***certificate*** that $\mathbf{mdbcs}_d(G)$ is also *big*.

(B) Otherwise, if **bw**(G) is ***small*** $(\leq \alpha \cdot \sqrt{k})$:
we compute $\mathbf{mdbcs}_d(G)$ efficiently using Catalan structures and ***dynamic programming*** techniques over an optimal branch decomposition of G.

Let us apply the general strategy...

We define the following **parameter** on a **planar** graph G:

$$\mathbf{mdbcs}_d(G) = \max\{|E(H)| \mid H \subseteq G \wedge H \text{ is connected } \wedge \Delta(H) \leq d\}.$$

(we focus on the **unweighted** version of the problem)

We distinguish two cases according to $\mathbf{bw}(G)$:

(A) If $\mathbf{bw}(G)$ is ***big*** $(> \alpha \cdot \sqrt{k})$:
we must exhibit a ***certificate*** that $\mathbf{mdbcs}_d(G)$ is also *big*.

(B) Otherwise, if $\mathbf{bw}(G)$ is ***small*** $(\leq \alpha \cdot \sqrt{k})$:
we compute $\mathbf{mdbcs}_d(G)$ efficiently using Catalan structures and ***dynamic programming*** techniques over an optimal branch decomposition of G.

Let us apply the general strategy...

We define the following **parameter** on a **planar** graph G:

$\mathbf{mdbcs}_d(G) = \max\{|E(H)| \mid H \subseteq G \wedge H \text{ is connected } \wedge \Delta(H) \leq d\}$.

(we focus on the **unweighted** version of the problem)

We distinguish two cases according to $\mathbf{bw}(G)$:

(A) If $\mathbf{bw}(G)$ is ***big*** $(> \alpha \cdot \sqrt{k})$:
we must exhibit a ***certificate*** that $\mathbf{mdbcs}_d(G)$ is also *big*.

(B) Otherwise, if $\mathbf{bw}(G)$ is ***small*** $(\leq \alpha \cdot \sqrt{k})$:
we compute $\mathbf{mdbcs}_d(G)$ efficiently using Catalan structures and ***dynamic programming*** techniques over an optimal branch decomposition of G.

Case (A)

Theorem (Robertson, Seymour & Thomas, 1994)

Let $\ell \geq 1$ be an integer. Every planar graph of branchwidth $\geq \ell$ contains an $(\ell/4 \times \ell/4)$-grid as a minor.

- Thanks to this result, it is enough to see:

(A.1) That the parameter is minor closed.

(A.2) How the parameter behaves on the square grid.

Case (A)

Theorem (Robertson, Seymour & Thomas, 1994)

Let $\ell \geq 1$ be an integer. Every planar graph of branchwidth $\geq \ell$ contains an $(\ell/4 \times \ell/4)$-grid as a minor.

- Thanks to this result, it is enough to see:

(A.1) That the parameter is minor closed.

(A.2) How the parameter behaves on the square grid.

Case (A)

Theorem (Robertson, Seymour & Thomas, 1994)

Let $\ell \geq 1$ be an integer. Every planar graph of branchwidth $\geq \ell$ contains an $(\ell/4 \times \ell/4)$-grid as a minor.

- Thanks to this result, it is enough to see:

(A.1) That the parameter is minor closed.

(A.2) How the parameter behaves on the square grid.

Case (A)

Theorem (Robertson, Seymour & Thomas, 1994)

Let $\ell \geq 1$ be an integer. Every planar graph of branchwidth $\geq \ell$ contains an $(\ell/4 \times \ell/4)$-grid as a minor.

- Thanks to this result, it is enough to see:

(A.1) That the parameter is minor closed.

(A.2) How the parameter behaves on the square grid.

Condition (A.1): the parameter is minor closed

Let G' be a minor of G.

- If G' occurs from G after an **edge removal**, then clearly $\mathbf{mdbcs}_d(G') \leq \mathbf{mdbcs}_d(G)$.
- If G' occurs after the **contraction** of an edge $\{x, y\}$: let $H' \subseteq G'$ be a solution, and let H be the *major* of H' in G

$\rightarrow$ We will show that we can find a connected subgraph $H^* \subseteq H' \subseteq G$ with $\Delta(H^*) \leq d$ and $|E(H^*)| \geq |E(H')|$.

Condition (A.1): the parameter is minor closed

Let G' be a minor of G.

- If G' occurs from G after an **edge removal**, then clearly $\mathbf{mdbcs}_d(G') \leq \mathbf{mdbcs}_d(G)$.
- If G' occurs after the **contraction** of an edge $\{x, y\}$: let $H' \subseteq G'$ be a solution, and let H be the *major* of H' in G

$\rightarrow$ We will show that we can find a connected subgraph $H^* \subseteq H' \subseteq G$ with $\Delta(H^*) \leq d$ and $|E(H^*)| \geq |E(H')|$.

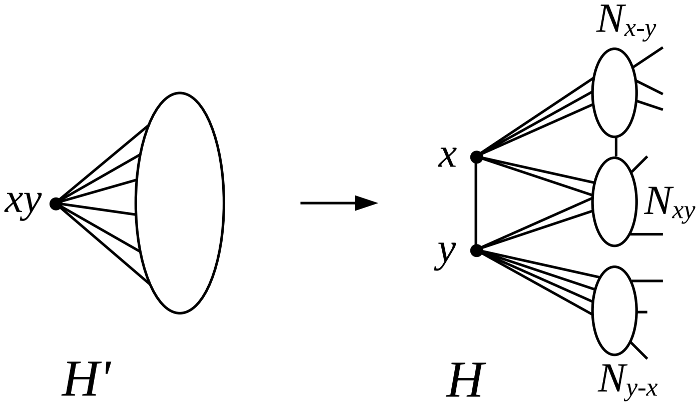
Condition (A.1): the parameter is minor closed

Let G' be a minor of G.

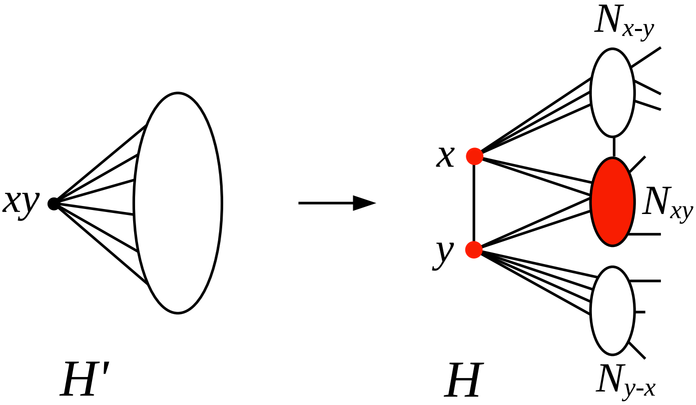
- If G' occurs from G after an **edge removal**, then clearly $\textbf{mdbcs}_d(G') \leq \textbf{mdbcs}_d(G)$.
- If G' occurs after the **contraction** of an edge $\{x, y\}$: let $H' \subseteq G'$ be a solution, and let H be the *major* of H' in G

$\rightarrow$ We will show that we can find a connected subgraph $H^* \subseteq H' \subseteq G$ with $\Delta(H^*) \leq d$ and $|E(H^*)| \geq |E(H')|$.

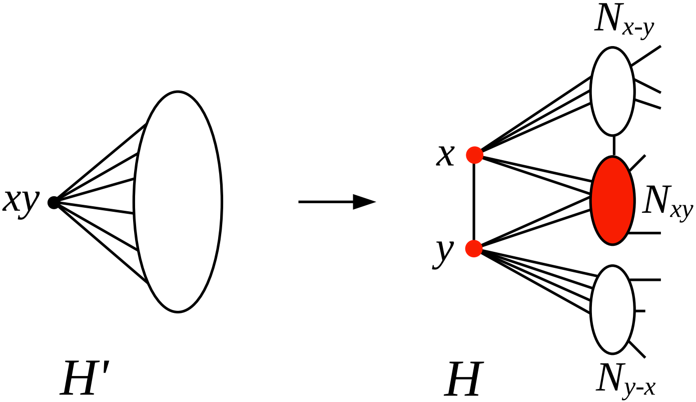
- $H' \subseteq G' \preceq_m G$.
- The edge $\{x, y\} \in E(G)$ has been contracted to the vertex $xy \in V(G')$.
- Let $H \subseteq G$ be the major of $H' \subseteq G'$.



- $N_H(x) \cup N_H(y) - \{x\} - \{y\} = N_{x-y} \sqcup N_{xy} \sqcup N_{y-x}$.
- x, y, and the vertices in N_{xy} may have degree $d+1$!!
- We will extract a subgraph $H^* \subseteq H'$ such that $|E(H^*)| \geq |E(H')|$. Suppose w.l.o.g. that $|N_{x-y}| \geq |N_{y-x}|$.



- If $|N_{x-y}| = d$, let $H^* = (V(H) - \{y\}, E(H) - \{x, y\})$.
- If $|N_{x-y}| < d$:
  - If $|N_{xy}| = 0$, let $H^* = H$.
  - If $N_{xy} = \{z_1\}$, let $H^* = (V(H), E(H) - \{x, z_1\})$.
  - If $N_{xy} = \{z_1, \ldots, z_k\}$ for some $k \geq 2$, let $H^* = (V(H), E(H) - \{x, z_1\} - \cup_{i=2}^{k}\{y, z_i\})$.



Condition (A.2): how it behaves in the square grid

- We must see that in an $(r \times r)$-grid R,

$$\mathbf{mdbcs}_d(R) = (\delta r)^2 + o((\delta r)^2).$$

- Indeed:
  - If $d = 2$, a Hamiltonian path in R gives

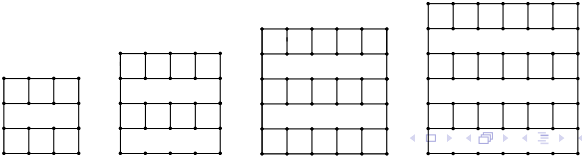
$$\mathbf{mdbcs}_2(R) \geq r^2 - 1.$$

  - If $d \geq 4$, the whole grid R is a solution, giving

$$\mathbf{mdbcs}_d(R) = 2r(r-1).$$

  - Finally, if $d = 3$, the subgraph below gives

$$\mathbf{mdbcs}_3(R) \geq 2r(r-1) - \left\lceil \frac{r-2}{2} \right\rceil (r-2).$$



Case (A): putting all together

Lemma (S. and Thilikos, 2008)

For any $d \geq 2$ and for any planar graph G it holds that

$$\mathbf{bw}(G) \leq \alpha \cdot \sqrt{\mathbf{mdbcs}_d(G)} + \mathcal{O}(1), \ \textit{with}$$

$$\alpha = \begin{cases} 4 & \textit{, if } d = 2 \\ 4\sqrt{2/3} & \textit{, if } d = 3 \\ \frac{4}{\sqrt{2}} & \textit{, if } d \geq 4 \end{cases}$$

Case (B): *fast* dynamic programming

Given an optimal *branch decomposition* (T, μ) of a planar graph G, there are 2 main ideas in the dynamic programming algorithm:

(B.1) Catalan structure in $\mathrm{mid}(e)$ to bound the size of the *tables*.

(B.2) How to deal with the connectivity in the *join/forget* operations.

Case (B.1): Catalan structures

- Given a set A, we define a *d-weighted partial partition* of A as any pair $(\mathcal{A}, \phi)$ where
  - $\mathcal{A}$ is a (possible empty) collection of mutually disjoint non-empty subsets of A, and
  - $\phi : A \to \{0, \ldots, d\}$ is a mapping corresponding numbers from 0 to d to the elements of A.
- Let $\mathscr{P}_e$ be the collection of all d-weighted partial partitions $(\mathcal{A}, \phi)$ of **mid**(e).
- We calculate $\textbf{opt}_e(\mathcal{A}, \phi)$ for each $(\mathcal{A}, \phi) \in \mathscr{P}_e$.
- If $|\textbf{mid}(e)| = \ell$ it is easy to see that $|\mathscr{P}_e| \leq f(\ell) \cdot (d+1)^\ell$, with $f(\ell) \leq 2^{\ell \cdot \log \ell}$.
- Can we say something better about $f(\ell)$??

Case (B.1): Catalan structures

- Given a set A, we define a *d-weighted partial partition* of A as any pair $(\mathcal{A}, \phi)$ where
  - $\mathcal{A}$ is a (possible empty) collection of mutually disjoint non-empty subsets of A, and
  - $\phi : A \to \{0, \ldots, d\}$ is a mapping corresponding numbers from 0 to d to the elements of A.
- Let $\mathscr{P}_e$ be the collection of all d-weighted partial partitions $(\mathcal{A}, \phi)$ of $\mathbf{mid}(e)$.
- We calculate $\mathbf{opt}_e(\mathcal{A}, \phi)$ for each $(\mathcal{A}, \phi) \in \mathscr{P}_e$.
- If $|\mathbf{mid}(e)| = \ell$ it is easy to see that $|\mathscr{P}_e| \leq f(\ell) \cdot (d+1)^{\ell}$, with $f(\ell) \leq 2^{\ell \cdot \log \ell}$.
- Can we say something better about $f(\ell)$??

Case (B.1): Catalan structures

- Given a set A, we define a *d-weighted partial partition* of A as any pair $(\mathcal{A}, \phi)$ where
  - $\mathcal{A}$ is a (possible empty) collection of mutually disjoint non-empty subsets of A, and
  - $\phi : A \to \{0, \ldots, d\}$ is a mapping corresponding numbers from 0 to d to the elements of A.
- Let $\mathscr{P}_e$ be the collection of all d-weighted partial partitions $(\mathcal{A}, \phi)$ of **mid**(e).
- We calculate $\mathbf{opt}_e(\mathcal{A}, \phi)$ for each $(\mathcal{A}, \phi) \in \mathscr{P}_e$.
- If $|\mathbf{mid}(e)| = \ell$ it is easy to see that $|\mathscr{P}_e| \leq f(\ell) \cdot (d+1)^{\ell}$, with $f(\ell) \leq 2^{\ell \cdot \log \ell}$.
- Can we say something better about $f(\ell)$??

Case (B.1): Catalan structures

- Given a set A, we define a *d-weighted partial partition* of A as any pair $(\mathcal{A}, \phi)$ where
  - $\mathcal{A}$ is a (possible empty) collection of mutually disjoint non-empty subsets of A, and
  - $\phi : A \to \{0, \ldots, d\}$ is a mapping corresponding numbers from 0 to d to the elements of A.
- Let $\mathscr{P}_e$ be the collection of all d-weighted partial partitions $(\mathcal{A}, \phi)$ of $\mathbf{mid}(e)$.
- We calculate $\mathbf{opt}_e(\mathcal{A}, \phi)$ for each $(\mathcal{A}, \phi) \in \mathscr{P}_e$.
- If $|\mathbf{mid}(e)| = \ell$ it is easy to see that $|\mathscr{P}_e| \leq f(\ell) \cdot (d+1)^{\ell}$, with $f(\ell) \leq 2^{\ell \cdot \log \ell}$.
- Can we say something better about $f(\ell)$??

Case (B.1): Catalan structures

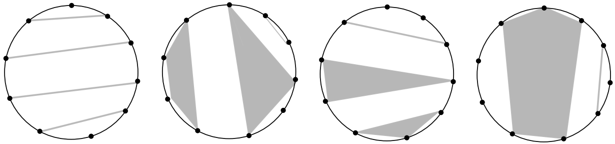
- Given a set A, we define a *d-weighted partial partition* of A as any pair $(\mathcal{A}, \phi)$ where
  - $\mathcal{A}$ is a (possible empty) collection of mutually disjoint non-empty subsets of A, and
  - $\phi : A \rightarrow \{0, \ldots, d\}$ is a mapping corresponding numbers from 0 to d to the elements of A.
- Let $\mathscr{P}_e$ be the collection of all d-weighted partial partitions $(\mathcal{A}, \phi)$ of $\mathbf{mid}(e)$.
- We calculate $\mathbf{opt}_e(\mathcal{A}, \phi)$ for each $(\mathcal{A}, \phi) \in \mathscr{P}_e$.
- If $|\mathbf{mid}(e)| = \ell$ it is easy to see that $|\mathscr{P}_e| \leq f(\ell) \cdot (d+1)^{\ell}$, with $f(\ell) \leq 2^{\ell \cdot \log \ell}$.
- Can we say something better about $f(\ell)$??

- *Sphere cut decomposition*: Branch decomposition where the vertices in **mid**(e) are situated around a cycle.
  $\rightarrow$ for any planar graph there exists an optimal branch decomposition which is also a sphere cut decomposition
  [P. Seymour and R. Thomas. Combinatorica'94]

- *Sphere cut decomposition*: Branch decomposition where the vertices in **mid**(e) are situated around a cycle.
  $\rightarrow$ for any planar graph there exists an optimal branch decomposition which is also a sphere cut decomposition
  [P. Seymour and R. Thomas. Combinatorica'94]

- *Sphere cut decomposition*: Branch decomposition where the vertices in **mid**(e) are situated around a cycle.
  → for any planar graph there exists an optimal branch decomposition which is also a sphere cut decomposition
  [P. Seymour and R. Thomas. Combinatorica'94]

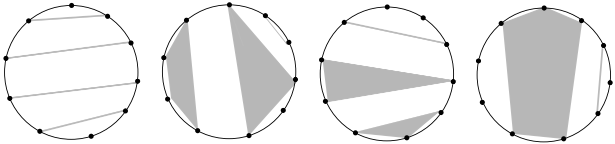
- We have to calculate in how many ways we can draw hyperedges inside a cycle such that they touch the cycle only on its vertices and they do not intersect:



- The number of such configurations is exactly the number of *non-crossing partitions* over ℓ vertices, which is equal to the ℓ-th *Catalan number* :

$$CN(\ell) = \frac{1}{\ell+1}\binom{2\ell}{\ell} \sim \frac{4^\ell}{\sqrt{\pi}\ell^{3/2}} \approx 4^\ell = 2^{\mathcal{O}(\ell)}.$$

- *Sphere cut decomposition*: Branch decomposition where the vertices in **mid**(e) are situated around a cycle.
  $\rightarrow$ for any planar graph there exists an optimal branch decomposition which is also a sphere cut decomposition
  [P. Seymour and R. Thomas. Combinatorica'94]
- We have to calculate in how many ways we can draw hyperedges inside a cycle such that they touch the cycle only on its vertices and they do not intersect:

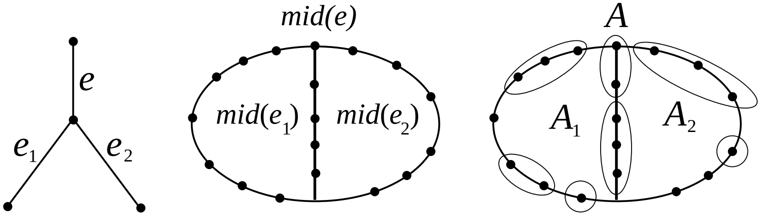


- The number of such configurations is exactly the number of *non-crossing partitions* over ℓ vertices, which is equal to the ℓ-th *Catalan number* :

$$CN(\ell) = \frac{1}{\ell+1}\binom{2\ell}{\ell} \sim \frac{4^\ell}{\sqrt{\pi}\ell^{3/2}} \approx 4^\ell = 2^{\mathcal{O}(\ell)}.$$

Case (B.2): How to deal with connectivity

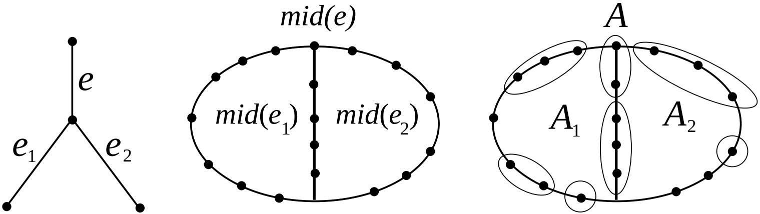
- **General idea**: we have to keep track of the connected components of the solutions, depending on how they intersect $\mathrm{mid}(e)$:



- We distinguish two cases according to the partition $\mathcal{A}$ of $\mathrm{mid}(e)$:
  - (1) $\mathcal{A} \neq \emptyset$.
  - (2) $\mathcal{A} = \emptyset$.

Case (B.2): How to deal with connectivity

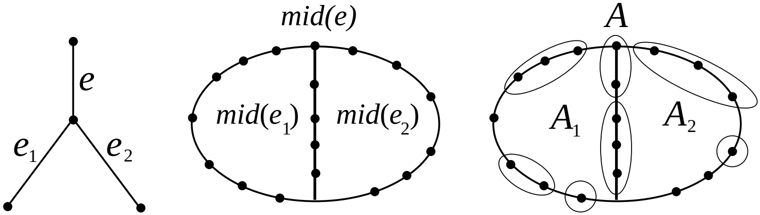
- **General idea**: we have to keep track of the connected components of the solutions, depending on how they intersect $\mathrm{mid}(e)$:



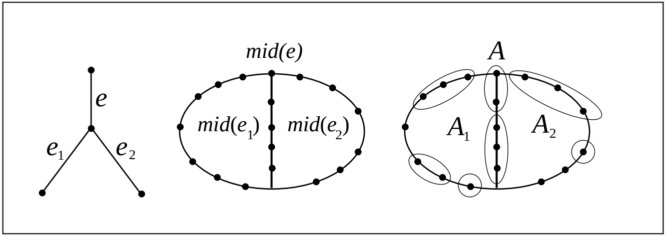
- We distinguish two cases according to the partition $\mathcal{A}$ of $\mathrm{mid}(e)$:
  (1) $\mathcal{A} \neq \emptyset$.
  (2) $\mathcal{A} = \emptyset$.

Case (B.2): How to deal with connectivity

- **General idea**: we have to keep track of the connected components of the solutions, depending on how they intersect $\mathrm{mid}(e)$:



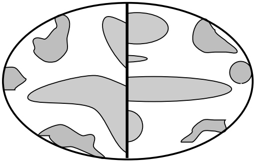
- We distinguish two cases according to the partition $\mathcal{A}$ of $\mathrm{mid}(e)$:
  (1) $\mathcal{A} \neq \emptyset$.
  (2) $\mathcal{A} = \emptyset$.



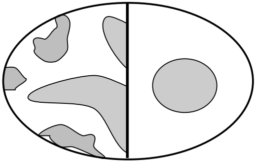
(1) Case $\mathcal{A} \neq \emptyset$.

(1.1) Case $\mathcal{A}_1 \neq \emptyset$, $\mathcal{A}_2 \neq \emptyset$.

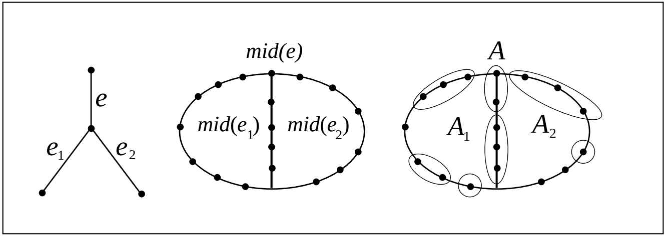
(1.2) Case $\mathcal{A}_1 \neq \emptyset$, $\mathcal{A}_2 = \emptyset$.



(1.1)



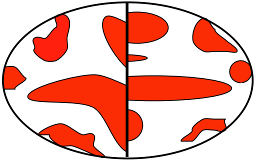
(1.2)



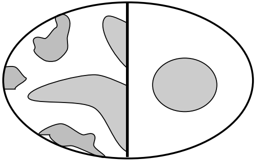
(1) Case $\mathcal{A} \neq \emptyset$.

(1.1) Case $\mathcal{A}_1 \neq \emptyset$, $\mathcal{A}_2 \neq \emptyset$.

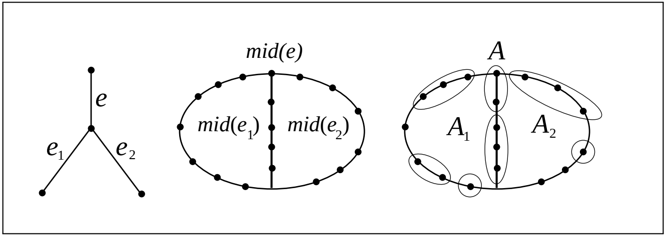
(1.2) Case $\mathcal{A}_1 \neq \emptyset$, $\mathcal{A}_2 = \emptyset$.



(1.1)



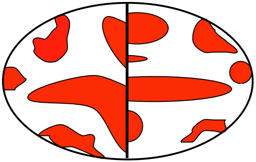
(1.2)



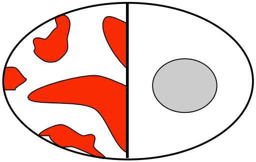
(1) Case $\mathcal{A} \neq \emptyset$.

(1.1) Case $\mathcal{A}_1 \neq \emptyset$, $\mathcal{A}_2 \neq \emptyset$.

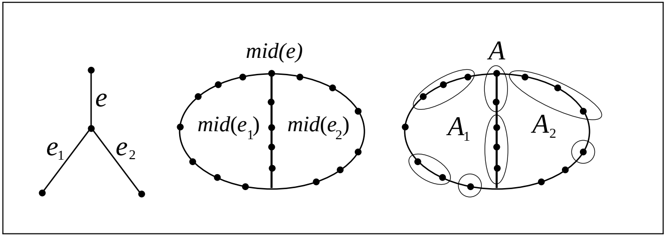
(1.2) Case $\mathcal{A}_1 \neq \emptyset$, $\mathcal{A}_2 = \emptyset$.



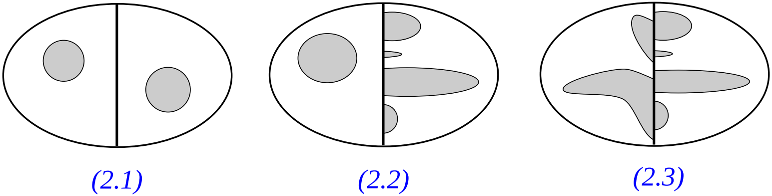
(1.1)

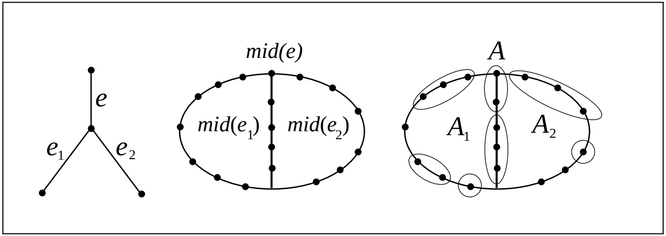


(1.2)



(2) Case $\mathcal{A} = \emptyset$.
- (2.1) Case $\mathcal{A}_1 = \emptyset$, $\mathcal{A}_2 = \emptyset$.
- (2.2) Case $\mathcal{A}_1 = \emptyset$, $\mathcal{A}_2 \neq \emptyset$.
- (2.3) Case $\mathcal{A}_1 \neq \emptyset$, $\mathcal{A}_2 \neq \emptyset$.



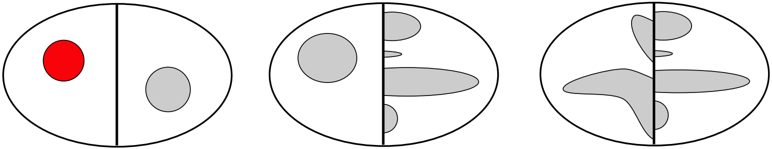


(2) Case $\mathcal{A} = \emptyset$.

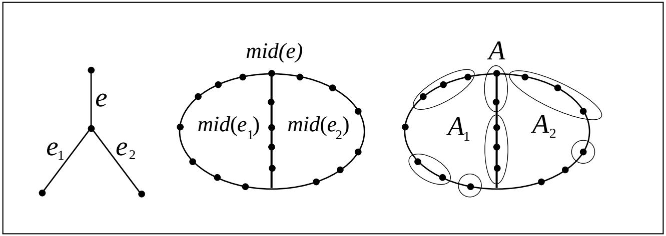
(2.1) Case $\mathcal{A}_1 = \emptyset$, $\mathcal{A}_2 = \emptyset$.

(2.2) Case $\mathcal{A}_1 = \emptyset$, $\mathcal{A}_2 \neq \emptyset$.

(2.3) Case $\mathcal{A}_1 \neq \emptyset$, $\mathcal{A}_2 \neq \emptyset$.



(2.1) *(2.2)* *(2.3)*

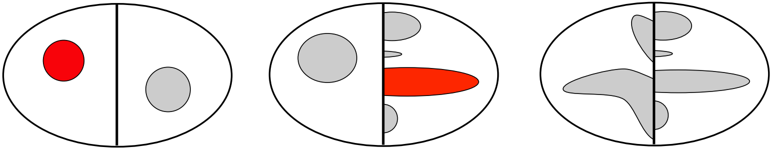


(2) Case $\mathcal{A} = \emptyset$.

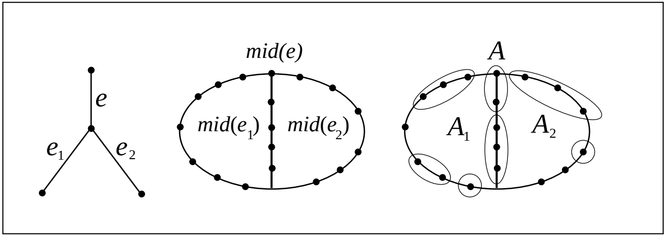
(2.1) Case $\mathcal{A}_1 = \emptyset$, $\mathcal{A}_2 = \emptyset$.

(2.2) Case $\mathcal{A}_1 = \emptyset$, $\mathcal{A}_2 \neq \emptyset$.

(2.3) Case $\mathcal{A}_1 \neq \emptyset$, $\mathcal{A}_2 \neq \emptyset$.

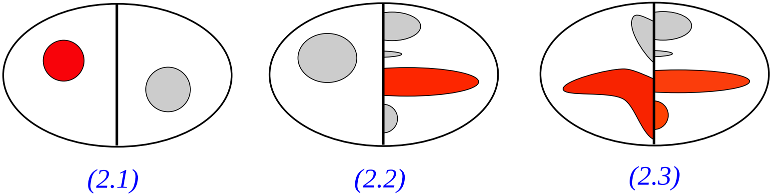


(2.1) (2.2) (2.3)



(2) Case $\mathcal{A} = \emptyset$.

(2.1) Case $\mathcal{A}_1 = \emptyset$, $\mathcal{A}_2 = \emptyset$.
(2.2) Case $\mathcal{A}_1 = \emptyset$, $\mathcal{A}_2 \neq \emptyset$.
(2.3) Case $\mathcal{A}_1 \neq \emptyset$, $\mathcal{A}_2 \neq \emptyset$.



Finally...

Theorem (S. and Thilikos, 2008)

k-PLANAR MAXIMUM d-DEGREE-BOUNDED CONNECTED SUBGRAPH *is solvable in time* $\mathcal{O}\left(2^{6\alpha\cdot\sqrt{k}}(d+1)^{3\alpha\cdot\sqrt{k}}n+n^3\right)$, *with*

$$\alpha = \begin{cases} 4 & \text{, if } d = 2 \\ 4\sqrt{2/3} & \text{, if } d = 3 \\ \frac{4}{\sqrt{2}} & \text{, if } d \geq 4 \end{cases}$$

- This strategy can be adapted to similar problems:
  - Maximising the number of vertices (instead of edges).
  - Replacing "*connected* subgraph" with "subgraph with *bounded* number of connected components".
  - ...

Finally...

Theorem (S. and Thilikos, 2008)

k-PLANAR MAXIMUM *d*-DEGREE-BOUNDED CONNECTED SUBGRAPH *is solvable in time* $\mathcal{O}\left(2^{6\alpha\cdot\sqrt{k}}(d+1)^{3\alpha\cdot\sqrt{k}}n+n^3\right)$, *with*

$$\alpha = \begin{cases} 4 & , \text{if } d = 2 \\ 4\sqrt{2/3} & , \text{if } d = 3 \\ \frac{4}{\sqrt{2}} & , \text{if } d \geq 4 \end{cases}$$

- This strategy can be adapted to similar problems:
  - Maximising the number of vertices (instead of edges).
  - Replacing "*connected* subgraph" with "subgraph with *bounded* number of connected components".
  - ...

Conclusions and further research

- We have described a framework to obtain **subexponential parameterized algorithms on planar graphs** for a family of problems dealing with degree-bounded connected subgraphs.

- There is still a loooooot of work to do:
  - Improve the running time.
  - Extend these algorithms to other sparse graph classes: bounded genus, minor-free, ...
  - Extend these algorithms to the edge-weighted version (one can prove that the parameter is still *minor closed*).
  - Consider a more general family of problems: largest subgraph excluding a fixed graph F as a minor...
  - ...

Conclusions and further research

- We have described a framework to obtain **subexponential parameterized algorithms on planar graphs** for a family of problems dealing with degree-bounded connected subgraphs.
- There is still a loooooot of work to do:
  - Improve the running time.
  - Extend these algorithms to other sparse graph classes: bounded genus, minor-free, ...
  - Extend these algorithms to the edge-weighted version (one can prove that the parameter is still *minor closed*).
  - Consider a more general family of problems: largest subgraph excluding a fixed graph F as a minor...
  - ...

Conclusions and further research

- We have described a framework to obtain **subexponential parameterized algorithms on planar graphs** for a family of problems dealing with degree-bounded connected subgraphs.
- There is still a loooooot of work to do:
  - Improve the running time.
  - Extend these algorithms to other sparse graph classes: bounded genus, minor-free, ...
  - Extend these algorithms to the edge-weighted version (one can prove that the parameter is still *minor closed*).
  - Consider a more general family of problems: largest subgraph excluding a fixed graph F as a minor...
  - ...

Conclusions and further research

- We have described a framework to obtain **subexponential parameterized algorithms on planar graphs** for a family of problems dealing with degree-bounded connected subgraphs.
- There is still a loooooot of work to do:
  - Improve the running time.
  - Extend these algorithms to other sparse graph classes: bounded genus, minor-free, ...
  - Extend these algorithms to the edge-weighted version (one can prove that the parameter is still *minor closed*).
  - Consider a more general family of problems: largest subgraph excluding a fixed graph F as a minor...
  - ...

Conclusions and further research

- We have described a framework to obtain **subexponential parameterized algorithms on planar graphs** for a family of problems dealing with degree-bounded connected subgraphs.
- There is still a loooooot of work to do:
  - Improve the running time.
  - Extend these algorithms to other sparse graph classes: bounded genus, minor-free, ...
  - Extend these algorithms to the edge-weighted version (one can prove that the parameter is still *minor closed*).
  - Consider a more general family of problems: largest subgraph excluding a fixed graph F as a minor...
  - ...

Gràcies!